A MULTI-LEVEL CONTROLLER SYSTEM

RELATED APPLICATIONS

This application claims the benefit, under 35 USC § 119(e), of U.S. Provisional Application No. 60/427,445 and U.S. Provisional Application No. 60/427,527, which are incorporated by reference herein.

FIELD OF THE INVENTION

The present invention relates generally to a state machine architecture and more specifically to a method and system for analyzing particles in a dilute fluid sample.

BACKGROUND OF THE INVENTION

Most processes consist of several process segments that fit together in a physically and temporally consistent manner. For example, a fluid control and imaging system, such as a urinalysis system, includes process segments for receiving a sample, aspirating the sample, and injecting the sample into a fluidics system where images are taken as the sample flows in a proper manner. To produce meaningful data, the sample receptacle, the aspirator, the valves, the pumps, and the optical components have to be in the proper positions at the proper times, and various fluids have to be released at the right time. Typically, these process segments are controlled on a real time basis by a single processor according to instructions in software or firmware. The instructions usually entail responding to various inputs by considering "hard-coded" branching conditions. This type of "hard-coded" branching takes the form of a "case statement" in which each case corresponds to a state and a relatively complicated "if" statement determines the response to inputs when in the given state.

In this type of real time operation, the process segments do not all happen completely simultaneously. For example, although the sample may be received and aspirated at the same time the flow cell is being flushed with a cleaning solution, the aspiration and the flushing take different lengths of time to complete, and therefore do not start and end at the same time. Furthermore, how long each of these processes will take to complete cannot always be predicted accurately because a process may have a random component (e.g., sample concentration will determine the time required to capture a fixed number of images) or a catastrophic unplanned event, such as the aspiration needle getting stuck or the cleansing fluid not being loaded correctly. Since the success of a process depends on each process segment running smoothly, this type of catastrophic unplanned event not only disrupts a process segment but also turns an entire process run into wasted effort if the event is not handled properly. Further, there is extra inefficiency associated with troubleshooting once it is determined that a run was erroneous because it is difficult to know exactly which process segment was in progress at a certain point in time.

In addition to the above disadvantages with a single-processor hard-coded real time controller, there is an additional source of inefficiency stemming from the need for a process engineer to collaborate with a software engineer to adjust the process parameters. While the process engineer understands the process and knows how the components ought to be set, implementation of the process usually requires a software engineer. Thus, typically, the process engineer has to explain the process and what he wants to accomplish to a programmer, who then revises the code. This process engineer – software engineer communication link not only takes time but also creates more opportunities for error based on miscommunication or misunderstanding.

For the above reasons, a method of controlling a process more efficiently is desired.

SUMMARY OF THE INVENTION

The invention is a system (e.g., a urinalysis system) that includes multiple levels of controllers. The system includes a first control level having at least one level-1 controller that moves through a sequence of first level states, the first control level generating a first level command that is associated with one of the first level states. The system also includes a second control level having at least one level-2 controller that moves through a sequence of second level states in response to the first level command. The level-2 controller sends a status report to the first control level when a level-2 condition that is defined in one of the second level states is fulfilled in the second level states.

The invention also includes a method of executing a process. The method entails identifying a sequence of first level states to be executed, issuing a first level command to a second level that has a level-2 controller, wherein the first level command is associated with one of the first level states, and receiving a second level status report from the level-2 controller. The status report indicates a status of the level-2 controller in response to the first level command.

In another aspect, the invention includes a method of controlling a system that includes controllers and system components. The method entails receiving input parameters in a first language that is not readable by the controllers, wherein the input parameters are instructions for controlling system components, converting the input parameters into translated parameters that are in a second language, wherein the second language is readable by the controllers, and creating a table containing the input parameters and corresponding translated parameters. The input parameters are editable in the table.

In yet another aspect, the invention is a system including a controller, system components, and a process for generating a table that is useful for controlling the controller and the system components. The table has a first set of columns containing instructions in a first language that is not readable by the controller, and a second set of columns containing instructions in a second language that is readable by the controller. The values and instructions in the second set of columns are translated versions of the instructions in the first set of columns.

In another aspect, the invention is a system comprising a level-1 controller that divides a level-1 task into a first level-2 task and a second level-2 task and issues a first level command to a first level-2 controller and a second level-2 controller, respectively, wherein the first level-2 controller executes the first level-2 task and the second level-2 controller executes the second level-2 task in response to the first level command, so that when the first and the second level-2 controllers complete their level-2 tasks, the level-1 controller completes the level-1 task.

The invention is also a multi-layered control system that includes a plurality of controllers in different control levels. Each controller behaves according to a controller table containing a unique set of values. Although the controller tables contain different values, the tables have substantially similar formats. Each controller table is indexed by states and has a first column of commands to issue to a lower level controller, a second column of status reports to send to a higher level controller, a third column of tests for checking whether a predefined condition is fulfilled, and a fourth column defining a course of action if the predefined condition is fulfilled. An interface control level receives commands from the plurality of controllers and controls system components in response to the commands.

BRIEF DESCRIPTION OF THE DRAWINGS AND APPENDICES

FIG. 1 is a flow chart of an exemplary urinalysis process that may be used with the invention;

FIG. 2 is a block diagram showing the main components of an exemplary urinalysis system;

FIG. 3 is a perspective view of an exemplary bench top module analyzer unit that may be used for the invention;

FIG. 4 is a schematic diagram of the analyzer unit;

FIG. 5 is a schematic presentation of the leveled system architecture in accordance with the invention;

FIG. 6 provides an exemplary set of run sequences that the host controller sends to the level-1 controller;

FIG. 7 demonstrates how the level-1 controller executes a run sequence once an activating signal is received from the host controller;

FIGs. 8A-8L (collectively "FIG. 8") depict exemplary tables that may be used for a level-1 controller in a urinalysis process; and

FIGs. 9A-9C ("FIG. 9"), 10A and 10B (FIG. 10), and 11A and 11B ("FIG. 11") depict exemplary tables that may be used for level-2 controllers in a urinalysis process.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The invention is particularly applicable to urinalysis processes and it is in this context that the invention will be described. It will be appreciated, however, that the system and method in accordance with the invention has greater utility since the system and method may be used with other types of processes that can be modeled as one or more state machines. The present invention may be incorporated into an in-vitro urinalysis diagnostic device that utilizes the technologies disclosed in U.S. Patents 4,338,024 and 4,393,466.

Generally, there are two types of state machines: synchronous state machines and asynchronous state machines. Synchronous state machines carry out segments of a process (or a time-slice of a process) synchronized to the beat of a clock. Thus, if a process were to be divided into a series of one-minute time slices, the pre-assigned tasks for each of those time slices are triggered at each clock signal. An advantage of a synchronous state machine is its stability, which stems from everything being synchronized to the clock. However, this synchronization can be a limitation of a synchronous state machine because synchronization often does not result in the fastest process runs. Furthermore, choosing the right clock speed can be tricky, for some parts of the process might want a faster clock speed than other parts. If the clock speed is too slow, certain events that start and finish between consecutive clock signals can get missed.

Asynchronous state machines provide a faster solution than the synchronous state machines. Asynchronous state machines, which do not operate to the beat of a clock, usually consist of a block of logic elements whose output is fed back as an input through a delay line. Thus, the output and next state of an asynchronous state machine is determined from a combination of the current state and the current input. The fact that the next state is immediately determined once the current state value transits the delay line allows the process to be run fast if the delay is shortened. However, these asynchronous state machines are not without a disadvantage. Since every change in the input test condition is immediately reflected at the output of the combinatorial logic and starts a new state propagating through the delay element, the asynchronous state machines can be unstable. Background information and details about asynchronous and synchronous state machines may be found in Tinder, Richard F., Engineering Digital Design (Academic Press, January 2000).

The invention offers a hybrid-type state machine that has the stability of the synchronous state machine and the speed of the asynchronous state machine but does not suffer the disadvantages of either. This hybrid-type state machine is implemented in the form of a multi-level controller system wherein one or more controllers may be present in each level. If there are multiple controllers in one level, the controllers in the same level behave independently of each other. The behavior of each controller is dictated by a state record formalism, which may be implemented in a spreadsheet/table. The spreadsheet has multiple fields, with each row being a "state record." Some fields test certain conditions, and the "next state" is identified based on the test result. The controller then transitions to the next state and carries out the values/commands of the fields in this next state.

Each controller in the multi-level controller system behaves according to its own spreadsheet. For example, for a system that includes one controller at the first control level and three controllers at the second control level, there would be four spreadsheets. The controller at the first control level issues commands to the three controllers at the second control level, the three controllers execute their tasks according to the commands, and send at least one signal back to the controller at the first control level when their tasks are complete. Although the three controllers do not necessarily finish their tasks at the same time, the controller at the first control level waits to hear back from all three controllers (or however many it needs to hear back from) before moving on to the next state in the spreadsheet for the first control level. Thus, while the three controllers at the second control level execute their tasks independently of each other, the controller at the first control level synchronizes the states to a degree by waiting to hear back from the three controllers before proceeding to the next state.

This hybrid-type state machine is like a synchronous state machine in that the controller at the first control level moves from a current state to a next state when a predetermined set of tasks are accomplished or conditions are met. However, this hybrid-type state machine is not restricted to the beat of a clock like a synchronous state machine. Rather, it can execute a process as fast as its components can execute their discrete tasks. By using faster processors, the hybrid-type state machine can be made to execute a process almost as fast as an asynchronous state machine.

The hybrid-type state machine also has characteristics of an asynchronous state machine. For example, it tests conditions to decide the next step, and as soon as these conditions are met it advances to the next state. However, it does not suffer the instability problem like an asynchronous state machine because the first level synchronizes the process segments at certain points in the process (i.e., at the end of each state), and because of the state record formalism all possible states and state transitions are specified. The state record formalism that is implemented in a spreadsheet with fields specifies all the parameters so that certain points of the process are synchronized.

Another advantage of this multi-level controller architecture is that the syntax at every level except the lowest level can be made the same, drastically simplifying the programming process. At each level in this multi-level controller system, the controller only "knows" and cares about what is in its spreadsheet; it does not know about all the details happening at the levels above or below its own level. For example, a three-leveled system may include 1) a high control level that executes run sequences, and 2) a middle control level that executes the states in each of the run sequences and issues commands to 3) a low control level that controls motors, actuators, and valves according to commands in the states. In this system, the high control level only knows to send a run sequence command to the middle control level, and to move to the next run sequence state once it hears back from all the controllers in the middle control level that it needs to hear back from. The high control level is unaware of all the details of the states in the middle control level, the existence of low control levels below the middle level, the function of the low control level and each of the valves and the motors, etc. Likewise, the middle control level only knows to send out certain commands to the low control level in response to a command from the high control level, and to determine the next state when it hears back from the low control level.

A benefit of using the spreadsheet is that if a process engineer's terminology (e.g., motor #2 on/off, test sensor #12, valve #3 on/off) is used in the fields of the spreadsheet, the process engineer can adjust the settings of a process run without getting help from a programmer. By inputting values into the fields of the spreadsheet(s), the process engineer can specify states and run sequences for each level by himself. The spreadsheet allows the process to be broken down into process engineering terminology by receiving the input parameters from the process engineer in the language of his preference, and converting these input parameters into a machine-usable code. The converted code may be resented in a section of the spreadsheet, such as a separate set of column(s).

As used herein, a "state" refers to a process segment that is associated with the controllers and the components of the system being in a certain position/setting or performing a specific task. The "task" can be instructions to do something active, such as opening a valve, or instructions to do something passive, like waiting. A row in the spreadsheets of FIGs. 8, 9, 10, and 11, for example, represents a state. A "state record" is a collection of the field values or parameters in a row of the spreadsheets in FIGs. 8, 9, 10, and 11. "Parameters" refer to values in a table. A state may be indexed by a "state index" or a "state order." The fields in a state record represent/specify the outputs from the controller when in the given state, along with the tests and conditions that these tests must satisfy to end the state. For example, if a level-1 controller is to issue first level commands to three controllers at a second control level, the level-1 controller will obtain the specific command for each of these level-2 controllers from the respective fields in its state record. These first level commands may identify the run sequence to be executed by each level-2 controller. In this case, each level-2 controller executes the second level states in the identified run sequence. As used herein, controllers in the first control level are referred to as "level-1 controllers," controllers in the second control level are referred to as "level-2 controllers," etc. Also, as used herein, "first level commands" refer to commands issued by a level-1 controller, "second level commands" refer to commands issued by a level-2 controller, etc. "Execution" of states means to move through the states in an appropriate order, taking appropriate actions in each state.

Multiple states make up a "run sequence." Typically, a run sequence has a specific set of states arranged in a specific order to achieve a specified function, such as sample aspiration or moving a specimen rack from point A to point B. Often, multiple state run sequences at the second control level corresponds to a single state in the first control level.

"Motors," as used herein, includes both motors that actually move mechanical parts and motors that drive pumps, etc. "Components," as used herein, refer to all physical portions of a system other than a controller. "Commands" are signals traveling from a controller at a higher level to a controller at a lower level, and "status reports" are signals traveling from a controller at a lower level to a controller at a higher level.

Now, the invention will be further described in reference to the Figures.

FIG. 1 is a flow chart of an exemplary urinalysis process 10 that may be implemented with the invention. The analysis process 10 begins when a urine sample is received (stage 12) by a sampling system. The received sample is mixed and aspirated into the system and introduced into a fluidics system (step 14). Elements in the flow microscope stain and mix the sample (step 16), then pass it through a flow cell (step 18) where images are taken. A host controller oversees the taking of the images. An imaging computer receives the data from the flow cell (step 20), processes them as needed (step 22), and displays the data (step 24).

FIG. 2 is a block diagram showing the main components of an exemplary urinalysis system 30. The urinalysis system 30 includes a sampler 32 and a microscopy analyzer unit 34, both of which feed information to a computer 36. A communication link 37 connects the microscopy analyzer unit 34 to the computer 36, which is also connected to a Laboratory Information System (LIS) 38. A specimen rack 40 that carries test tubes of samples travels between the sampler 32 and the microscopy analyzer unit 34. In some embodiments, an automatic rack transfer mechanism transfers the specimen rack 40 from the sampler 32 to the microscopy analyzer unit 34. In other embodiments, the transfer is done manually.

FIG. 3 is a perspective view of an exemplary bench top module microscopy analyzer unit 34 that may be used for the invention. The microscopy analyzer unit 34 aspirates samples, collects images from urine samples using digital image capture of analyte images presented in a flow microscope, and performs image processing to isolate individual particles. The microscopy analyzer unit 34 has its own power supply, processor, and controllers, a barcode reader to identify samples, motors to drive the mechanical components of the system, fluidics systems to pipette samples from the test tubes, and a communication link 37 to the computer 36 (see FIG. 2). On the outside, the microscopy analyzer unit 34 includes a platform 42 onto which the specimen rack 40 is placed, and some buttons, e.g., a start button 44. The specimen rack 40 is loaded on a measurement side 44 of the platform 42. Multiple (e.g., five) racks may be loaded at the same time. An input conveyor (CI) moves the rack away from the user and toward a wall 48, where the samples are aspirated by a pipette system (not shown). As each sample is aspirated, a shifter arm incrementally moves the rack toward a return side 46, placing the next sample test tube under the pipette. When all the test tubes are sampled, an output conveyor (CO) near the return side 46 moves the rack forward, away from the wall 48. Where multiple racks are placed on the platform 42, the rack that was at the very back (i.e., near the wall 48) on the measurement side 44 will end up at the front (i.e., away from the wall 48) on the return side.

FIG. 4 is a schematic diagram of the microscopy analyzer unit 34. The nine connectors 50 along the left hand side of the figure are connected to a level-1 controller 52 (HLC) and a level-2 controller 54 that is controlled by the level-1 controller. In the embodiment shown in FIG. 4, the level-2 controller includes a first level-2 controller 54a (also called the specimen presentation assembly (SPA) controller), a second level-2 controller 54b (also called the fluid block assembly (FBA) controller), a third level-2 controller 54c (also called the specimen transport mechanism (STM) controller), and a fourth level-2 controller 54d (also called the optical Block Assembly (OBA) controller). Each of the level-2 controllers controls certain system components.

The first level-2 controller 54a controls the aspirator process by sending signals to pipette motors 60, 62, an evacuation pump 64, a sheath pump 67, and some of the valves 68. A pipetting station controlled by the first level-2 controller 54a is located near the front of the microscopy analyzer unit 34. The pipetter, controlled by the pipette motors 60, 62, mixes the sample and aspirates a predetermined amount of the sample. The sample is mixed in its tube, a sampling probe descends to the tube bottom and expels a pulse of air to assure uniform mixing of sediment before being processed by the microscopy analyzer unit 34. After the mixing, the sample is drawn by vacuum into a flowcell through a series of tubes that allow stain to be introduced into and mixed with the urine.

The second level-2 controller 54b controls the flow cell by sending signals to a cannula pump 70, the sheath pump 67, and some of the valves 68. In the particular embodiment, the valves 68 are controlled by both the first and the second level-2 controllers 54a and 54b. The fluidics components are composed of a stain container (not shown), a sheath container (not shown), and three pumps 64, 67, and 70 that aspirate and circulate the sample, the stain, sheath fluid, and a cleaner (e.g., IRISolv) into the fluid block and the flow cell. The pumps 64, 67, and 70 may be peristaltic pumps or displacement pumps.

The third level-2 controller 54c sends signals to components that function to transport the specimen rack 40, by sending signals to various arms that move the rack along a path. More specifically, the third level-2 controller 54c sends signals to a front arm stepper motor 76a, a rear arm stepper motor 76b, one or more DC motors 78, to move the specimen rack 40. Sampling from the tubes on the sample platform 42 (shown in FIG. 3) is performed by a pump-driven fluidic subsystem. Any suitable pump, including a displacement pump of the type described in U.S. Patent Application Serial No. ________ [Attorney Docket No. 2102402-914911], may be used to drive the sampling process.

The third level-2 controller 54c interfaces a barcode reader 90. The barcode reader 90 scans the barcode label on a sample tube and keeps the identification information stored locally. Eventually, the barcode reader 90 sends the scanned identification information to the Laboratory Information System (LIS) 38 via the computer 36.

The fourth level-2 controller 58 controls the optical components of the flow microscope, which may include motorized positioners 86 within a microscope (not shown) to adjust the position of a flow cell. In the embodiments in which the microscopy analyzer unit 34 utilizes the strobe illumination, the strobe flashes at a high speed, synchronized with the CCD camera 84. The CCD camera 84, which is controlled by the host controller, captures the image during the strobe flash. The level-1 controller indirectly controls the strobe bulb 82. The microscope typically includes components such as diffusion filters, lenses, and other optical elements to focus the light in a desired manner. In the embodiment shown, some of these components are controlled by the fourth level-2 controller 54d.

After the images are taken, the waste pump 94 discards the fluids into a waste line that leads to the waste chamber. A waste well liquid level sensor assembly 73 detects the liquid level in the waste chamber.

FIG. 5 is a schematic presentation of the leveled system architecture 100 in accordance with the invention. Unlike in a conventional system, which executes real time operation using a single processor, the leveled system architecture 100 includes multiple levels of processors and/or controllers that control each other and ultimately the system components. For example, the host level controller 56 includes a host processor, the level-1 controller 52 includes the master controller, the level-2 controller 54 includes four controllers

54a, 54b, 54c, and 54d, and the level-3 controller includes one or more controllers that interface between the level-2 controllers and the system components. The motors, pumps, and the valves are controlled by the level-3 controllers, which are programmed in C and typically are not changeable like the state records, which a process engineer accesses to control the operations of the second and the level-2 controllers 52, 54.

In the embodiment shown, the host level controller 56 includes an Analysis Processor (AP) 56a and a Sediment Module Host (SMH) 56b. The SMH 56b directly controls the level-1 controller 52. The user interface, however, is implemented in the AP 56a. Thus, from the user's perspective, there are two paths for controlling the system: 1) push buttons on the microscopy analyzer unit 34 (see FIG. 3), or 2) enter commands using the user interface (e.g., a GUI) in the AP 56a. The two paths allow the user to control different aspects of the system. The invention is not limited to the particular configuration of the host level controller 56 shown in FIG. 5.

In this multi-level system architecture 100, the "higher" levels send commands to the "lower levels," which perform certain functions according to the commands and return at least one status report to the "higher" levels that issued the commands. For example, the level-1 controller 56 sends first level commands to the level-2 controller 52, and the level-2 controller 52 sends second level commands to the level-3 controller 54 in accordance with the first level commands from the level-1 controller 56. The level-3 controller 54 performs its functions according to the second level commands from the level-2 controller 52, and returns a status report to the level-2 controller 52 when the functions are completed. The level-2 controller 52 waits to receive all the status reports it expects to receive and, when this happens, sends a status report to the level-1 controller 56 to inform the level-1 controller that the level-2 controller 52 completed its functions. "Completion," as used herein, can mean either a successful completion of the prescribed task(s) or a passage of a prescribed amount of time.

This leveled system architecture 100 may be used with the urinalysis process 10 described above, although its utility is not so limited. Although not shown in FIGs. 8, 9, 10, and 11, each of the spreadsheets contains a column with a header file generated by the spreadsheet for programming the controller. The header file usually takes the form of a hexadecimal file containing the state record information, and becomes loaded into an EEPROM associated with each of the microcontrollers that implement the level-1 controllers 52 and the level-2 controllers 54. In this way, the process engineer, who does not have to be a programmer, can input field values into the spreadsheet(s), which will automatically generate the code interpreted by the controllers. By entering or modifying values in the spreadsheets for the level-1 controller 52 and the level-2 controller 54, the process engineer can modify the process parameters without relying on a programmer.

FIG. 6 provides an exemplary set of run sequences that the host controller 56 sends to the level-1 controller 52. In the embodiment shown, each run sequence is identified by a two-letter code L_1L_2 600. For each run sequence, there is a beginning state 602, an end state 604, and a general description 606 of what is done in the particular run sequence. For example, in run sequence RH, which is carried out in states 1 through 14, the level-2 controllers 54 are reset. In run sequence 16, which is carried out in states 15 through 99, the system resumes running racks. There is no limitation to the order in which the run sequences are executed, and not all the run sequences need to be executed during a process. Also, some states may be executed in multiple run sequences. For example, Run Sequence PI for "Irisolve Clean," which includes states 350-366, is part of Run Sequence WA for "wash with bleach" (states 356-392). The Sediment Module Host PC portion of the host controller 56 will send signals for Run Sequence PI or Run Sequence WA depending on what the process engineer wants done.

FIG. 7 is a state machine record interpreter that demonstrates how the level-1 controller 52 and each other level except the lowest level controller executes a run sequence once an activating signal is received from the host controller 56 (or the next higher level controller). Initially, upon activation, the level-1 controller 52 is reset (stage 700) and is in an idle state (stage 702). At this point, the current state is the state identified in the "beginning state" column in FIG. 6. After the functions of the first state are completed, the controller proceeds to the subsequent stage in the sequence and adjusts the state number (stage 704) so that a state that was "the next state" in the previous state is now "the current state." The controller waits for the subsequent stage to be completed (stage 706), then moves on to the next stage as shown by an arrow 708. After repeating stages 704 and 706 until there is no next state in the sequence, at which point the run sequence is completed and the level-1 controller 52 returns to the idle state in stage 702.

FIGs. 8, 9, 10, and 11 depict exemplary tables that may be used to implement the urinalysis process 10. The table in FIG. 8 is a level-1 controller state record table 300, which dictates the behavior of the level-1 controller 52. The tables in FIGs. 8, 9, and 10 are level-2 controller state record tables that dictate the behavior of the level-2 controller 54. The tables are arranged so that a state is represented by each row and a signal is represented by each column. In the tables, "SM" refers to the controller or processor at an adjacent higher level. For example, SM from the perspective of a level-1 controller 52 would be the host controller 56, and SM from the perspective of a level-2 controller 54 would be the level-1 controller 52.

FIG. 8 is an exemplary level-1 controller state record table 300. The state record table 300 is in the form of a spreadsheet with columns indicating different fields and rows indicating different states. The level-1 controller 52 moves from a current state to a next state until the run sequence has been completed. A state index 302 shows a numerical designation for a state that is described in a description column 304. For each state, there are commands 306 that the level-1 controller 52 issues to the level-2 controller 54. In response to the commands 306a, 306b, 306c, and 306d, the level-2 controllers 54a, 54b, 54c, and 54d perform certain functions and return status reports 308a, 308b, 308c, and 308d to the level-1 controller 52. The level-2 controllers 54 execute their functions independently and asynchronously with respect to one another. Since the level-2 controllers 54 may take different lengths of time to complete their respective functions, they will each return their respective status reports to the level-1 controller 52 at different points in time. The level-1 controller 52 does not send a command to execute the next state until all the expected status reports have been returned. There is no restriction to the number of status reports that the level-1 controller 52 needs to receive before moving on to the next state. Values in the status report fields 308 can be changed by a process engineer, and in table 300, "0X00" means no status report is needed.

In addition to the fields 306, 308 that pertain to communicating with the level-2 controllers 54, the level-1 controller state record table 300 contains field parameters pertaining to communicating with the host controller 56. SMTest 310 refers to a signal that the level-1 controller 52 receives from the host controller 56. A particular value of the SMTest 310 stands for a two-letter code, which is interpreted differently depending on the state of the level-1 controller at the time the two-letter code is received. For example, if the level-1 controller is idle at the time a particular code is received, the level-1 controller will interpret the code as a run sequence shown in FIG. 6. Thus, it will execute the run sequence after accessing the table of FIG. 6 to identify the run sequence. It also sends a signal ToSM 312 to the host controller to inform the host controller that the run sequence is in progress. Some time later, the host controller will issue another two-letter code to stop the run sequence, for example because all the necessary data have been collected. This time, since the level-1 controller is running at the time the two-letter code is received, the level-1 controller checks to see if the received two-letter code matches the value in the SMTest 310 field. If the value matches the value in the SMTest 310, the level-1 controller will end the state. The level-1 controller will then send a signal ToSM 312 to indicate that the run sequence has ended.

Tvalue 314 and Tfunc 316 together determine how long a state lasts. More specifically, Tvalue 314 indicates a time frame in some predetermined temporal unit and Tfunc 316 indicates how the time frame indicated in the Tvalue 314 is to be used (e.g., as the maximum run time). If a state is not completed within the time frame defined by Tvalue 314 and Tfunc 316, an error message may be generated to alert the user that an unexpected event happened in this state.

Sens 318, STst 320, and SMsk 322 pertain to determining the sensor state. There are a number of sensors in the microscopy analyzer unit 34 for measuring or detecting a parameter, and Sens 318 identifies the sensor that is relevant to a particular state. STst 320 identifies the status of the sensor (e.g., on or off). SMsk 322 provides a reference value that is compared against the STst value to determine a course of action. The STst value having a certain relationship with respect to the reference value is herein referred to as a "condition" for taking the subsequent step toward completion.

The signal from the host controller (SMTest 310), the statuses 308 of the third level controllers, and the status of the sensor (STst 320) relative to Smsk 322 determine the next state. If these values indicate that no branching is to occur, the current state ends and the next consecutive-numbered state index in the current run sequence becomes activated. If the current state ends at the end of a run sequence, a ToSM 312 signal is sent to the host controller 56 to indicate that a run sequence has been completed. The level-1 controller 52 will then remain idle until the host controller 56 issues another SMTest signal, activating a new run sequence. If SMTest 310, the level-2 controller statuses 308, and the sensor status STst 320 indicate that the current state is to branch, thereby satisfying the "predefined condition" for branching, Dest 328 points to the state to branch to and Bran 326 triggers the branching.

As mentioned above, a process engineer is able to adjust at least some of the parameter values in table 300 because they are parameters that make sense when the system is considered from the perspective of a process engineer. For example, the process engineer may adjust the Tvalue 314 so that a state waits for a longer or shorter period of time before generating an error signal, or adjust values in commands 306 so that a certain valve is open, not closed. The system includes a user interface through which the process engineer can adjust "calibration" values in separate tables (not just the level-1 controller table). These calibration values may be used to control time intervals, motor operation or as threshold settings for sensor readouts.

The controllers in the urinalysis system 30 convert the parameters input by the process engineer to generate the machine-readable values in column 330. Although a programmer is initially needed to program this conversion method, the programmer does not need to be involved each time a parameter is adjusted.

When the level-2 controllers 54a, 54b, 54c, and 54d receive commands 306 from the level-1 controller 52, each of the level-2 controllers 54 starts a function in accordance with the received command. As mentioned above, the spreadsheets for the level-2 controllers 54a, 54b, 54c, and 54d have substantially the same syntax as the spreadsheet for the level-1 controller. This similarity extends not only to the fields in the spreadsheets but also to the mode of communication with the adjacent levels. For example, in the same way that the level-1 controller's interpretation of a code from the adjacent higher level depends on the state of the level-1 controller at the time the code is received, the level-2 controller interprets an input differently depending on whether it is in an idle state or an active state when the code is received.

FIG. 9 illustrates what the first level-2 controller 54a does upon receiving a command from the level-1 controller 52. A first level-2 controller state record table 400 contains the commands to be sent to the valves 68, the pipette motors 60, 62, an evacuation pump 64, and the sheath pump 67 (see FIG. 4) in columns labeled Energized Valves 406, First Motor 410, Second Motor 412, EP 408, and SP 414, respectively. The positions of the valves and pumps are indexed by a state order 402, each of which has a description 404. The "SP" here is a dummy field that is not used in the particular embodiment, as indicated by a constant value in all the states.

The first level-2 controller 54a uses a pressure sensor to measure the pressure inside the air storage. This air is used to mix the specimen, for example by air-blasting. Sensor Select 416 identifies the sensor, Sensor State 418 indicates the status of the selected sensor, and Sensor Mask 420 contains a reference value. MotorTest 422, which detects the status of the motor, is one of the factors considered to determine what to do next. A motor may be in one of a few states, the exact number depending on the motor type. For example, a motor may be 1) running and not where it is supposed to be, 2) running and where it is supposed to be, or 3) stopped where it is supposed to be. The value in the MotorTest 422 indicates which of the three states the motor is in.

Unlike the level-1 controller 52, the first level-2 controller 54a does not test for specific messages from a higher-level controller, which in this case would be the level-1 controller 52. Thus, SM Test 424 is set at a value 0X0000. However, the statuses of the valves and motors are reported to the level-1 controller, as indicated by non-zero values in ToSM 426. Tvalue 428 indicates a time out value associated with each respective state and Tfunc 430 indicates how the time out value is to be used. Based on the value in MotorTest 422 (i.e., whether they satisfy the predefined condition with respect to the reference value), Tvalue 428, and Tfunc 430, the first level-2 controller determines whether the state is to end or not. If the state is to end, the value in the column labeled End Ctrl 432 indicates how the particular state is to end. If the state is to branch, a Bran Ctrl 434 briggers the branching and a Dest 436 points the state or the run sequence to branch to.

FIG. 10 depicts a second level-2 controller table 500 that may be used to direct the second level-2 controller 54b to execute certain functions. The second level-2 controller table 500 is indexed by a state order 502, each of which has a State Description 504. The second level-2 controller table 500 is similar to the first level-2 controller table 400 (see FIG. 4 above) in that it controls valves 68. In addition, it controls the sheath pump 67 and the cannula pump 70. MotorTest 522 detects the status of the sheath pump 67 and the cannula pump 70. Energized Valve 506 identifies the fluid block valves that are to be energized. Command columns 509 include a CP 508, which indicates a signal to be sent to the cannula pump, and an SP 514, which indicates a signal to be sent to the sheath pump. Dummy Motor 510 and Dummy Motor 512 are unused in the embodiment shown, as indicated by a constant value 80FF in both columns. The Dummy Motor fields may be used if more motors are added to the system. Likewise, the columns relating to sensor selection, such as Sensor Select 516, Sensor State 518, and Sensor Mask 520 are unused in the embodiment shown because the second level-2 controller does not use a sensor. The unused fields may be used if a component, such as a sensor, is used with the second level-2 controller 54. The motor states are tested and the status codes are compared to the test value in Motor Test 522. There is a two-way communication from the second level-2 controller 54b, the level-1 controller 52, as shown by the signals in ToSM 526. There is no signal transmitted from the level-1 controller 52 to the second level-2 controller 54b, however, as indicated by the constant value 0X0000 in SMTest 524.

Tvalue 528 indicates a time out value associated with each respective state and Tfunc 530 indicates how the time out value is to be used. If the current state does not end because the predefined conditions in EndCtl are not satisfied before the amount of time indicated by the time out value expires, an error signal may be generated. Depending on whther values of Motor Test 522, Tvalue 528, and Tfunc 530 satisfy this predefined condition or not, the state will either end or branch. The value in the column labeled End Ctrl 532 indicates the conditions to be satisfied for the particular state to end. If the state is to branch, a Bran Ctrl 534 indicates a condition or manner of branching and a Dest 536 points the state or the run sequence to branch to. In state 24, Dest = 0, indicating that the next state is an idle state (state order 0).

FIG. 11 depicts a third level-2 controller state record table 600 for the third level-2 controller 54c, which controls the specimen rack 42 (see FIG. 3). The third level-2 controller state record table 600 is indexed by state order 602 and a description 604. As described above in FIG. 3, there are at least two conveyors in the system: an input conveyor (CI) and an output conveyor (CO). CI pushes the specimen rack toward the wall 48 of the microscopy analyzer unit 34 (see FIG. 3). The rack then shifts sideways incrementally to allow sampling of the different test tubes. A shifter arm (SM), controlled by the stepper motor 76a in FIG. 4, shifts the specimen rack 40 from the CI to the CO along this path, stopping to allow each tube to be sampled. Then, the CO moves the specimen rack from the position where CI left it and brings the specimen rack to the front, so that CI and the CO together take the specimen rack on a U-shaped path. A repositioning arm (CM) controlled by the stepper motor 76b (in FIG. 4) could be used to send a command 612 to move the rack from the finishing end of the U to the beginning end of the U so that the CI is able to move the specimen rack again. SM 610, CM 612, CI 614, and CO 616 show signals to be sent to the respective arms in each state. In the exemplary table 600, the columns Dor 606 and Dand 608, and RC 618 are dummy columns, as indicated by a constant value. These dummy columns may get used if additional components are incorporated into the system.

A few sensors are used by the third level-2 controller 54c. SenSel 620 identifies a sensor to be tested in each state, Stst 622 indicates the status of the sensor as determined by the test, and SMtst 624 indicates a reference value that is used for testing the status of the sensor. Mtst 626 is the encoded status of motors 76a, b, or 78a, b (see FIG. 4). SMtst 628 is unused, indicating that the level-1 controller 52 is not tested. Signals in ToSM 630 are sent to the level-1 controller 52 when certain conditions are fulfilled.

Tvalue 632 indicates a time out value associated with each respective state index and Tfunc 634 indicates how the time out value is to be used. Depending on whether the values of SM 610, CM 612, CI 614, CO 616, and Mtst 626 satisfy the predefined condition or not, the current state will either end or branch. If the state is to end, the value in the column labeled End Ctrl 636 indicates how the particular state is to end. If the state is to branch, a Bran Ctrl 638 triggers branching and a Dest 640 points the state or the run sequence to branch to.

Example

This example illustrates how the multi-level controllers interact with each other to complete a run sequence. Specifically, this example illustrates how the level-1 controller resets the level-2 controllers and has them execute run sequences that prepare the system for processing samples (i.e., execute the first run sequence shown in FIG. 6). As shown in FIG. 6, the run sequence RH entails state indices 1-14 of the level-1 controller state record table 300. During this process, the sheath bottle is filled with a sheath fluid, the pipette position is initialized, and the STM initializes the position of rack motion levers and clears the input and output conveyor. In the first state of this reset process (i.e., FIG. 8, state index = 1), the level-1 controller 52 sends a reset signal (RE) to each of the four high level controllers. The four level-2 controllers 54 reset themselves in response to the reset signal, and send back a status report (0XFF). After receiving all four status reports, the level-1 controller 52 sends the signals associated with state index 2. In state index 2, the level-1 controller 52 sends a signal "S1" to the first level-2 controller 54a. The first level-2 controller 54a, which will then start state 61 of the state record table 400, fills the sheath bottle. Since the value "0X0000" indicates that no signal is sent, no signal is sent to the second, third, and fourth level-2 controllers in state index 2. When the sheath bottle starts to get filled, the first level-2 controller 54a returns a completion status "0X32," as shown in the ToSM column of table 400 and the SPAstat column of table 300. The value 0x0002 in the End column 324 indicates that state index 2 will end as soon as the "0x32" status is returned from first level-2 controller 54a.

State index 3, which has a timeout value 314 of 0x0005 and a Tfunc 316 of 0x42, says to stop filling the sheath bottle when the time out value 314 is reached. The Sens 318 value of 0x0033, STst value of 0x01, and SMsk value of 0x01 indicate that if the sensor identified by 0x0033 reaches a state 0x01, the sheath bottle is full. The End column 324 has a value 0x8383, which means that the state is to end when either the sheath bottle is full or the timeout value 314 is reached. When the current state ends, the level-1 controller proceeds to state index 4, whereby an S0 signal is sent to the first level-2 controller 54a to turn off the filling. Upon receiving the signal S0, the first level-2 controller 54a turns off the sheath pump and sends the signal 0X33 to the level-1 controller 52, which then ends the current state.

In state index 5, the level-1 controller 52 waits for the first level-2 controller 54a to send a signal 0XFF indicating that the sheath pump is turned off and it is now idle.

After the sheath pump is filled, the sheath's level is tested in state index 6. The sheath level is tested by identifying the low sensor selected by 0X0032 in column 318, and comparing it against the calibration table value selected by SMsk value 0x80. If the comparison indicates that the sheath level is low, the process branches to state index 8 according to the Bran 32 and the Dest 8. In state index 8, it is tested whether the sheath is empty. If the sheath sensor 0X0032 in column 318 indicates that the sheath is empty when compared against the calibration table value selected by SMsk value of 0x40, the Bran 326 value and the Dest value 328 direct the level-1 controller to branch to state index 10. In state index 10, the level-1 controller sends a message 0x21 to the host processor to let the host processor know that the sheath is empty. Then, the level-1 controller branches to state index 100, which starts the rack clearing process.

Referring back to state index 6, if the sensor selected by 0X0032 does not indicate that the sheath level is low, no branching occurs and the next state would be state index 7. In state index 7, the level-1 controller 52 reports that the sheath level is okay to the host processor by sending a ToSM message 0x1F. Then, the level-1 controller 52 executes state index 11.

In state index 11, the level-1 controller 52 sends the commands PH, HS, and CR to the first level-2 controller 54a, the second level-2 controller 54b, and the third level-2 controller 54c, respectively. No signal is sent to the fourth level-2 controller (the OBA controller) because optical bench initialization is not required in the reset run sequence. The first level-2 controller, upon receiving the PH command, starts to lift the pipetter and send the pipetter to a back sensor and eventually to a waste well by executing states 4-9 in table 400. In states 4 and 5, signals are sent to the horizontal motor to move the pipette arm to the back sensor. In state 6, a signal is sent to the motor to rotate the pipetter out to the waste well. Then, air pressure is checked in state 7 by using a sensor 10B0 to compare against a reference sensor mask value. If the pressure is at an acceptable level already, the SPA controller 36b branches to state 9, where a message 0x12 is sent to the level-1 controller to inform that the pressure level is acceptable. If the pressure is not at an acceptable level, the first level-2 controller 54a sets valves and activates the air pump to recharge the pressure to the acceptable level in state 8 before sending the signal to the level-1 controller in state 9.

The second level-2 controller 54b, upon receiving the HS command, activates the valve CBV3 and sends the signal 80FE to the cannula pump 70 and the sheath pump 67 (see FIG. 4) to put them in standby positions in state 32. The pump roller is then placed on the tube in state 33. Then, a status report signal 0X21 is sent to the level-1 controller 52 in state 34.

The third level-2 controller 54c, upon receiving the CR command, runs the infeed conveyor in reverse by sending a signal 0X80FF to the SM, 0X0100 to the CI, and 0X880FF to the CO. This is done for a time period defined by Tval = 0X0016 and Tfnc = 0X02. After this predefined time period, a signal 0x30 is sent to the level-1 controller and state 34 ends.

Referring back to table 300, an FBAstat value of 0x21 is received in state 11. A Tvalue of 0x0009 and a Tfunc value of 0x44 set a time frame by which the FBAstat value must be returned. The End value 0x005F indicates that if the FBAstat value of 0x21 is not received by the end of the designated time frame, an error signal is generated; if the FBAstat value is received in time, the level-1 controller moves on to state 12.

In state index 12, the level-1 controller 52 waits to receive a SPAstat value of 0xFF and moves on to state 13 when it does. In state 13, the level-1 controller waits to receive an STMstat value of 0xFF, and moves on to state 14 when it does. In the run sequence Reset High (RH), no signal is exchanged between the level-1 controller and the OBA controller. In state 14, the Reset High run sequence ends.

With the leveled controller architecture 100 (see FIG. 5) of the invention, each level-2 controller 54 operates independently of each other. However, because the level-1 controller 52 does not proceed to the next state until all the expected status reports have been received from the high level controllers, the level-1 controller synchronizes the high level controllers at certain "check points" in the process. If not all the expected status reports are received within a designated time frame (e.g., Tvalue), an error signal is generated and, optionally, the process halts. Having this "check point" system at the end of each state prevents the entire process run from being tainted with an error. Furthermore, the "check point" system facilitates troubleshooting by allowing the process engineer to know exactly in which state the error occurred.

While the foregoing has been with reference to a particular embodiment of the invention, it will be appreciated by those skilled in the art that changes in this embodiment may be made without departing from the principles and spirit of the invention, the scope of which is defined by the appended claims.

For example, it is within the scope of the invention to allow the higher control levels to control a lower control level or the system components "directly" or "indirectly." If the control is "direct," there is no intervening level between the level that is doing the controlling and the level that is being controlled. For example, a level-1 controller usually directly controls a level-2 controller. In contrast, if the control is "indirect," there are one or more intervening levels between the level that is doing the controlling and the level that is being controlled. For example, in a three-level control system, the first control level may be designed so that it is able to control either the second control level or the third control level, depending on the situation.

In the embodiment including multiple lower level controllers, a command issued by a higher level controller is meant for, or "targeted to," one or more of the specific lower level controllers. In some embodiments, the higher level controller sends a command only to the targeted lower level controllers. In these embodiments, the lower level controllers that are not targeted do not receive the command. In other embodiments, the higher level controller "broadcasts" a command to many of the lower level controllers and let recipient lower level controllers determine whether the command applies to them. If a lower level controller determines that the command is not applicable, it will ignore the command.

Depending on the embodiment, controllers in the same control level are allowed limited or full communication with each other. Sometimes, the communication is limited to inter-level communication between a higher level controller and a lower level controller. A person of ordinary skill in the art will appreciate that numerous variations are possible.